PATENT
Customer No. 22,852
Attorney Docket No. 09851.0006-00000

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	
Anil Singhal, et al.	Group Art Unit: 2132
Application No.: 10/637,431	Examiner: Jung W. Kim
Filed: August 8, 2003	Confirmation No.: 2626
For: INTRUSION DETECTION SYSTEM AND NETWORK FLOW DIRECTOR METHOD	

FILED ELECTRONICALLY

Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

SECOND DECLARATION UNDER 37 C.F.R. § 1.131 ("Declaration B")

1. I, Danny Lobo, also known as Dionisio Lobo, and I, Anil Singhal, hereby declare:

2. Dionisio Lobo is a named inventor in the above-referenced application ("the Application"). He is also Director of Engineering of assignee NetScout Systems, Inc. of Westford, Massachusetts (NetScout).

3. Anil Singhal is named inventor in the Application. He is also Founder, President, CEO, and Chairman of the Board of assignee NetScout.

4. The Application claims priority to a U.S. provisional patent application, entitled INTRUSION DETECTION SYSTEM AND METHOD, Serial No. 60/402,255 ("the Provisional").

Approved By: RL Date: 2/1/08

5. Concurrently herewith, we have submitted a Declaration Under 37 C.F.R. § 1.131 ("Declaration A"), to establish a conception of the invention disclosed in the Provisional and claimed in at least the independent claims (claims 1, 21, 40, and 42) in the Application prior to July 30, 2002 and certain of the dependent claims (at least claims 2, 5-7, 15, 24-27, 41 and 43) that are embodiments of the independent claims, and to establish diligence in reducing the invention so claimed to practice between prior to July 30, 2002 and August 9, 2002. This Declaration Under 37 C.F.R. § 1.131 ("Declaration B") is also being submitted to establish a conception of the invention disclosed in the Provisional and claimed in the Application prior to July 30, 2002, and to establish diligence in reducing the invention to practice from before July 30, 2002 to August 9, 2002. Specifically, this Declaration addresses the subject matter claimed in dependent claims 3-4, 8-12, 13-14, 22-23, 28, and 32-34.

I. EVIDENCE OF CONCEPTION BEFORE JULY 30, 2002

6. Prior to July 30, 2002, in Westford, MA, we (Danny Lobo and Anil Singhal) conceived the invention disclosed in the Provisional and claimed in the Application. More particularly, prior to July 30, 2002, in Westford, MA, we conceived of a method for intrusion detection as recited in independent claim 1 that further features:

- "receiving, at a probe, data packets communicated over a third network link", as recited in claim 3;
- "aggregating the data packets received over the first network link and the data packets received over the third network link", as recited in claim 4;
- "maintaining, by the probe, an audit trail buffer for forensic analysis", as recited in claim 8;
- the audit trail buffer of claim 8 comprising "a memory for recording monitored packets", as recited in claim 9;

- the memory of claim 9 "record[ing] packets from at least one of the first network link and the third network link", as recited in claim 10;
- "receiving, by the probe, an event notification; and upon receipt of the event notification, communicating, by the probe, the current contents of the audit trail buffer", as recited in claim 11;
- "storing received packets in a collection buffer; stripping header information associated with a protocol of the first network link; and adding header information associated with a protocol of the second network link.", as recited in claim 12;
- the storing step of claim 12 comprising "storing packets received from at least one of the first network link and a third network link", as recited in claim 13; and/or
- "the stripping step [of claim 12] further compris[ing] stripping header and checksum information associated with a protocol of the first network link; and the adding step [of claim 12] further compris[ing] adding header and checksum information associated with a protocol of the second network link", , as recited in claim 14.

7. Prior to July 30, 2002, in Westford, MA, we also conceived of a network performance probe system as recited in independent claim 21 that further features:

- "a third network interface for monitoring packets communicated over a third network link", as recited in claim 22;
- "an aggregator for aggregating the packets from the first network link and the packets from the third network link", as recited in claim 23;
- "a performance analyzer for acquiring network performance data in response to the monitored packets communicated over the first network link", as recited in claim 28;
- "an audit trail buffer maintainable for forensic analysis", as recited in claim 29;
- the audit trail buffer of claim 29 comprising "a memory for recording monitored packets for forensic analysis", as recited in claim 32;

- "an event notification receiver for causing the probe, upon receipt of the event notification, to communicate the current contents of the audit trail buffer", as recited in claim 33; and/or
- "a collection buffer for storing received packets; a stripper for stripping header information associated with a protocol of the first network link; and an adder for adding header information associated with a protocol of the second network link", as recited in claim 34.

8. Based on our invention and prior to July 30, 2002, we prepared, in the U.S., computer software code, portions of which and printouts of the output for which are attached as Exhibits I - R. These Exhibits demonstrate, through use of a software program time stamp attached here as Exhibit G, conception of the above-described inventions before July 30, 2002.

9. The code that we prepared and that is documented in Exhibits G and I - R is an implementation of one embodiment of the claimed invention. It comprises a NetScout intrusion detection system operating in conjunction with a NetScout probe model 9200 to detect any malicious network traffic and network usage which cannot be detected by a conventional firewall.

10. The software programs from which the Exhibits have been extracted contain instructions implementing several functionalities that are neither relevant nor useful to implement the invention claimed in the Application. Only those portions of the software code that are necessary to show the conception of the claimed inventions have been attached as Exhibits hereto; the remainder of the software code has been blocked from its respective Exhibit to preserve its confidentiality. Because the Exhibits are multi-page documents, when an entire page of the Exhibit was blocked, the entire page was deleted from the Exhibit. Also, in accordance with MPEP 715.07.II, dates have been blocked from the Exhibits in order to preserve confidentiality. We declare that the dates are all prior to July 30, 2002.

11. Although other persons contributed to the development of the multiple functionalities implemented in the larger software code, portions of which were extracted and are attached as Exhibits I - R, Danny Lobo and Anil Singhal were responsible for the preparation of the computer software code that implemented the claimed invention, either by preparing the code themselves or having the code prepared at their direction.

A. <u>A third network interface for monitoring packets communicated over a third network link (Claim 22)</u>

12. As shown in Exhibit I, the code includes instructions for providing "a third network interface for monitoring packets communicated over a third network link", as recited in claim 22. Exhibit I consists of portions of a software program entitled DRVCFG.C, which contains instructions to implement the network performance probe system recited in claim 22. Page 1 of Exhibit I shows that DRVCFG.C bears a copyright notice in the name of Frontier Software Dev. Inc., which was the name of NetScout at the time that the software program DRVCFG.C was created. DRVCFG.C identifies the driver "FEC", the driver "CC3i", and the device "IDS." As established in Declaration A, the WAN link CC3i is an embodiment of the first network link and the Ethernet driver FEC is an embodiment of a second network link, such as recited in independent claim 1. DRVCFG.C identifies the Model 9200 probe as being IDS capable to allow the IDS device to work as the third network link. The IDS device operates as an embodiment of the third network link over which data packets may be transported.

See, for example, Code Fragment 1 of Exhibit I (page 3):

```
#elif (MODEL_9200 && !_IDS)
#define FEC     1

#elif (MODEL_9200 && _IDS)
#define FEC     1
#define CC3i        1
```

B. **Receiving, at a probe, data packets communicated over a third network link (Claim 3)**

13. As shown in Exhibits I and J, the code includes instructions for "receiving, at a probe, data packets communicated over a third network link", as recited in claim 3. DRVCFG.C, described above as Exhibit I, has instructions, "#elif (MODEL_9200 && !_IDS)" and "#elif (MODEL_9200 && _IDS)", that show that the Model 9200 probe is capable of operating in IDS and non-IDS mode. The non-IDS mode is a receive-only mode, to allow the probe to receive data packets from sources such as the IDS device. Exhibit J consists of portions of a software program entitled FECMAIN.C, which implements the Fast Ethernet driver FEC. FECMAIN.C is invoked by the data packet converter code to do the function of transmitting/forwarding the data packets over the IDS device (the third network link) in communication with the FECMAIN.C driver (the second network). The description of the function, "FecRecvFrame", in FECMAIN.C shows that data packets are received. See, for example, Code Fragment 2 of Exhibit J (page 41):

```
/*************************************************************************
 * Function:    FecRecvFrame.                                            *
 * Description: Receive Packet Handler.                                  *
 *              If NO pkts are received then info->frame_size will be    *
 *              set to ZERO.                                             *
 *                                                                       *
 * Returns:     XMIB_FRAME_INFO struct.                                  *
 *************************************************************************/
 EXPORT XMIB_FRAME_INFO  *FecRecvFrame (ifn)
```

14. The comments in FECMAIN.C indicate further that data packets in the form of frames are received by the Model 9200 probe from, for example, an IDS based device. See also, for example, Code Fragment 3 of Exhibit J (page 43):

```
    // Check if frame has arrived.
    // A frame can arrive on the High Priority Q or on the Low Prority Q.
    // If non-partial xfr mode is used, frames will arrive on LoPri Q only
    // In partial xfr mode, the header will arrive on HiPri Q and trailer
    // will arrive on LoPri Q.
    // Note:  Hi Pri Q <=> Q2   and   Lo Pri Q <=> Q1

  // Check the Completion Q1.
```

```
// Note: In partial xfr mode, only 1 receive completion Q (i.e. Q1) is
// used.
SF_READ_REG(CompletionQ1ProducerIndex, &CompletionQ1ProducerReg);
SF_READ_REG(CompletionQ1ConsumerIndex, &CompletionQ1ConsumerReg);

RxComQProducerIndex = (N16)
    CompletionQ1ProducerReg.b.RxCompletionQ1ProducerIndex;
RxComQConsumerIndex = (N16)
    CompletionQ1ConsumerReg.b.RxCompletionQ1ConsumerIndex;

if (RxComQConsumerIndex != RxComQProducerIndex)
    { // Frame has arrived
```

C. **Aggregating the data packets received over the first network link and the data packets received over the third network link (Claims 4, 23)**

15. As shown in Exhibit K, the code includes instructions for "aggregating the data packets received over the first network link and the data packets received over the third network link", as recited in claim 4. Exhibit K also contains instructions for features in claim 23, which, though of different scope from claim 4, recites similar elements and was rejected under the same rationale. Exhibit K consists of portions of a software program entitled REALCOL.C, which contains network performance evaluation code. REALCOL.C provides for processing data packets received from both the WAN link driver CC3i (the first network link) and the IDS device (the third network link), and for aggregating the data packets so received. REALCOL.C has a flag, "LINKAGGR", that demonstrates that aggregation of data is supported, and an instruction, "CALL PP PROCESS (MIRROR_IFN);", that calls a mirror interface to process the packet from both the WAN link driver CC3i and the IDS device. See, for example, Code Fragment 2 of Exhibit K (pages 6 - 7):

```
#elif (_FEC) || (_LINKAGGR)
    /*
    ** For FEC:  do this if only Fast Ether Channel option is on
    ** For 8700, 87CC: do this only if Link Aggrgation is on
    */
    if ((info->mirr_ifn_enabled == TRUE) && (info->ifn > 2))
            {
            /* save info ptrs. */
            UINT vifn  = info->vifn;
```

```
        UINT lifn  = info->lifn;
        UINT lvifn = info->logical_vifn;

        info->vifn = info->lifn = info->logical_vifn = 0;
        CALL_PP_PROCESS(MIRROR_IFN);

        /* retrieve the saved info ptrs. */
        info->vifn = vifn;
        info->lifn = lifn;
        info->logical_vifn = lvifn;
        }
#endif // model 73xx, 8200, 8700, 87cc
```

D. Maintaining, by the probe, an audit trail buffer for forensic analysis (Claims 8, 29)

16. As shown in Exhibit L, the code includes instructions for "maintaining, by the probe, an audit trail buffer for forensic analysis", as recited in claim 8. Exhibit L also contains instructions for features in claim 29, which, though of different scope from claim 8, recites similar elements. Exhibit L consists of portions of a software program entitled COLCAPT.C, which is used to capture and buffer data that could be used for forensic analysis and as an audit trail. The comment of COLCAPT.C, "Capture related structures", show that data are captured, and the entry, "CaptureBufferEntry", shows that data are buffered. The buffers that are developed by COLCAPT.C are re-usable buffers of a limited number. Once exhausted, they can be replenished again. These buffers contain data directly pertaining to the operation performed. See, for example, in Code Fragment 1 of Exhibit L (page 2):

```
/*
** Capture related structures.
** All Packet indices are one-based.
*/

typedef struct
      {
      CaptureBufferEntry_HDR  hdr;
      TRACE_BUCKET_HDR *first_bucket;      /* first logical bucket ptr   */
      TRACE_BUCKET_HDR *curr_bucket;       /* current used bucket ptr    */
      TRACE_BUCKET_HDR *ret_bucket;    /* bucket ptr, used in retrieve */
      N32          count_bucket;     /* no. of buckets in list     */
      N32          packet_index;     /* last packet in the buffer  */
      N32          capture_start_time;
                              /* time (ms) capture started  */
```

```
  N32          max_bufsize;     /* Max buffer size allowed    */
N32      bucket_size;
  /* These fields are used for allocate/commit/release operations. */
  TRACE_BUCKET_HDR *new_first_bucket;
  N32 new_bucket_count;
  }  CAPTURE_ROOT;
```

17. COLCAPT.C contains "if-then" instructions for analyzing the captured data. See, for example, Code Fragment 5 of Exhibit L (page 10):

```
TRACE(RETRIEVE_CAPTURE);

      if (key_len != key_len);       /* unreferenced argument */

      if (key_type == KEY_TYPE_CAPTURE_HDR)
            return(&cr->hdr);

      if ((key_type != KEY_TYPE_CAPTURE_DATA) &&
            (key_type != KEY_TYPE_FAST_CAPTURE_DATA))
            {
            log_event("retrieve_capture: invalid key type %#0.8lx",
                                                key_type);
            return(0);
            }

      /* check if capture has started ? */

      if (cr->hdr.CapturedPackets == 0)
            return(0);

      if (get_next)
            getnext_id(cr, key_ptr);

      id = GET_BE_32((TRN32 *) key_ptr);

      if (key_type == KEY_TYPE_FAST_CAPTURE_DATA)
            return( fast_retrieve_capture(cr, id) );
```

E. <u>The audit trail buffer comprising a memory for recording monitored packets (Claims 9, 32)</u>

18. As also shown in Exhibit L, the code of .COLCAPT.C includes instructions for providing "the audit trail buffer compris[ing] a memory for recording monitored packets", as recited in claim 9. Exhibit L also contains instructions for features in claim 32, which, though of different scope from claim 9, recites similar elements and was rejected under the same

rationale. The comments of COLCAPT.C show that it contains instructions for allocating a CAPTURE root data base for storing data. The call, "col_activate_capture", operates to initialize memory allocation logic. The call, "malloc", is an operating system call for requesting memory which is then used for capture logic. See, for example, Code Fragment 2 of Exhibit L (pages 5 - 6):

```
EXPORT BOOL col_activate_capture(handler_id, max_entry)
      UINT handler_id;
      UINT max_entry;
      {
      CAPTURE_ROOT *cr;

      TRACE(COL_ACTIVATE_CAPTURE);

      /* Allocate a CAPTURE root database. */
      if ((cr = malloc(sizeof(CAPTURE_ROOT))) == 0)
            return(FALSE);

      /*
      ** Invalidate all entries to force OP_CONFIG_COLLECTION to be done
      */
      memset(cr, 0, sizeof(CAPTURE_ROOT));
      cr->hdr.operation = OP_CONFIG_COLLECTION;
      cr->packet_index = 1;
      cr->max_bufsize = ((N32) max_entry) << 10;

      /* Set pointers to collection and retrieval handler functions. */

      handler_list[handler_id].collector = collect_capture;
      handler_list[handler_id].retriever = retrieve_capture;
      handler_list[handler_id].writer    = write_capture;
      handler_list[handler_id].deactivate = deactivate_capture;
      handler_list[handler_id].stats_ptr = cr;

      return(TRUE);
      }
```

F. **The memory recording packets from at least one of the first network link and the third network link (Claim 10)**

19. As also shown in Exhibit L, the code includes instructions for providing the "memory record[ing] packets from at least one of the first network link and the third network link", as recited in claim 10. COLCAPT.C contains instructions for recording packets used for

forensic analysis and for maintaining an audit trail. The instruction reciting "wanhdr_enabled" show that the packets are recorded from the WAN link or CC3i driver. See, for example, Code Fragment 3 of Exhibit L (page 6 - 7):

```
PRIVATE VOID collect_capture(cr, frame_ptr, frame_size,
                        frame_status, frame_time, frame_id)
      CAPTURE_ROOT *cr;
      N8  *frame_ptr;
      UINT frame_size;
      N32 frame_status;
      N32 frame_time;
      N32 frame_id;
      {
      N32 pkt_time;
      CaptureBufferEntry *p;
      WAN_MACHDR *wmac = 0;
      ATM_HDR *AtmHdrPtr = 0;
      VLAN_INFO *vlan_info = 0;
      BOOL wanhdr_enabled;
      BOOL PassAtmHeader;
      BOOL PassVlanHdr;
      N8 *dp;
      UINT n, n1;
      N16 size;

      TRACE(COLLECT_CAPTURE);

      if (cr->hdr.operation != OP_START_COLLECTION)
            return;             /* Nothing to do, if capture not on */

      wanhdr_enabled = PassAtmHeader = PassVlanHdr = FALSE;
```

20. During the invocation of COLCAPT.C, the captured packets are stored in the buffer entry in the memory that was allocated as was described above in paragraph 18. See, for example, Code Fragment 4 of Exhibit L (page 8):

```
/* Now save the buffer in the bucket */

      if ( (p = (CaptureBufferEntry *) capt_get_bufp(cr->curr_bucket)) == 0 )
            {
            log_event("collect_capture: NULL buffer ptr provided !\n");
            return;
            }

      p = (CaptureBufferEntry *) ((N8 *) p + cr->curr_bucket->space_used);
```

G. **An event notification receiver for causing the probe, upon receipt of the event notification, to communicate the contents of an audit trail buffer (Claims 11, 33)**

21. As shown in Exhibit M, the code includes instructions for "receiving, by the probe, an event notification; and upon receipt of the event notification, communicating, by the probe, the current contents of the audit trail buffer", as recited in claim 11. Exhibit M also contains instructions for features in claim 33, which, though of different scope from claim 11, recites similar elements and was rejected under the same rationale. Exhibit M consists of portions of a software program entitled RETPDIR.C, which is a component of an event notification receiver. Exhibit M shows instructions for response to an RMON event. The call, "HANDLER RMON EVENT", initiates the event notification that will invoke the function, "ret_init_event." Similarly, the call, "HANDLE RMON CAPTURE", when initiated, will invoke the function "ret_init_capture" for communicating the current contents of the audit trail buffer . See, for example, Code Fragment 1 of Exhibit M (page 26):

```
case HANDLER_RMON_CAPTURE:
      bp = (BufferControlEntry *) p;
      bp->packet = calloc(PACKET_BUF_SIZE,1);
      if (bp->packet == 0)
            {
            mib_control_del(p);
            return(0);
            }
      ret_init_capture(p);
      break;

case HANDLER_RMON_EVENT:
      ep = (EventEntry *) p;
      ep->Community = 0;
      ep->Description = calloc(sizeof(N32)
                        + MAX_EVENT_DESC_LEN+1,1);
      if (ep->Description != 0)
            ep->Community
                  = calloc(1,sizeof(N32)
                        + MAX_EVENT_COMMUNITY_LEN+1);
      if ( (ep->Description == 0) || (ep->Community == 0) )
            {
            mib_control_del(p);
            return(0);
            }
      ret_init_event(p);
```

```
break;
```

H. **A collection buffer for storing received packets (Claims 12, 34), the packets received from at least one of the first network link and the third network link (Claim 13)**

22. As shown in Exhibits J, N, and O, the code includes instructions for "storing received packets in a collection buffer", as recited in claim 12. Exhibits J, N, and O also contain instructions for features in claim 34, which, though of different scope from claim 12, recites similar elements and was rejected under the same rationale. Further, the instructions include instructions for the storing step to comprise "storing packets received from at least one of the first network link and a third network link", as recited in claim 13. The Ethernet driver identified as FECMAIN.C (Exhibit J above) contains instructions to implement a buffer for storing received data packets. FECMAIN.C includes the call "AllocateAdaptermemory", which allocates memory for the collection buffer for the network link. See, for example, Code Fragment 1 of Exhibit J (page 31):

```
// Allocate memory for descriptors+ buffers for both Rx and Tx
if (!(AllocateAdapterMemory(Adapter)))
            return (EDRV_CONFIG_FAIL);

// Initialize some variables in the adapter structure
Adapter->AvailableTxDesc = SF_NUMBER_OF_TX_DESC;
Adapter->PhysicalSegmentThreshold = SF_PHYSICAL_SEGMENT_THRESHOLD;
Adapter->TxBuffersInUse = 0;
Adapter->TxBufferIndex = 0;
Adapter->DPCQueued = FALSE;
Adapter->ResetInProgress = FALSE;
Adapter->RxGfpNoResponseInt = FALSE;
```

23. Exhibit N consists of portions of a software program entitled FECHW.C, which, in conjunction with FECMAIN.C, implements a collection buffer for storing received packets. The program defines the above-described memory-allocating call, "AllocateAdaptermemory." See, for example, Code Fragment 3 of Exhibit N (page 7):

```
/*****************************************************************************
 * Function:    AllocateAdapterMemory                                        *
 * Description: Allocates memory required for:                               *
 *      - Transmit descriptors and completion queue                          *
 *      - Transmit buffers (copy buffers for fragmented packets              *
 *      - Receive descriptors and completion queue                           *
 *      - Receive buffers and their flush buffers                            *
 *      - Structures to map xmt ring entries back to the packets             *
 *                                                                           *
 * Returns:     TRUE, if successful.                                         *
 *              FALSE, if failure.                                           *
 *****************************************************************************/

EXPORT BOOL AllocateAdapterMemory (Adapter)
      SF_ADAPTER  *Adapter;
{
UINT ii;

      //
      // Allocate Memory for XMT
      //
#ifdef ORIGINAL
UINT PoolBuffersNeeded;
N32  TmpAddress;

   // We need some NDIS_BUFFERs to describe memory that we
   // allocate (generally either to flush the buffer, or
   // because we need its physical address). To do this
   // we need a buffer pool, so we have to determine how
   // many buffers we need.
   PoolBuffersNeeded = SF_NUMBER_OF_RX_DESC + SF_MAX_TX_BUFFERS;

   NdisAllocateBufferPool(
        &AllocStatus,
        &Adapter->FlushBufferPoolHandle,
        PoolBuffersNeeded);

   if (AllocStatus != NDIS_STATUS_SUCCESS) {
      return FALSE;
   }
#endif
```

24. Exhibit O consists of portions of the software program entitled CC3iAPi.C, which implements the WAN link driver CC3i, which, as noted above, is an embodiment of the first network link. CC3iAPi.C contains instructions for receiving the packets from the WAN LINK CC3i. The routine "cc3i_recv_frame" is the corresponding routine which performs this function. See, for example, Code Fragment 1 of Exhibit O (page 16):

*

```
** Receive the next frame if any from the specified port.
** Note : In the current implementation, if a frame size > recv buffer size
**      and the frame is spread in more than one buffer than only the
**      first buffer will have proper data, while the remaining data
**      copied by upper layers will be garbage. !!
*/
EXPORT XMIB_FRAME_INFO *cc3i_recv_frame(ifn)
      UINT ifn;
      {
      XMIB_FRAME_INFO *info;
      RXDESCR *fd;
      IOCB *iocb;
      UINT datacnt;
      INT  inuse_port;
      BOOL discard;
      N8   frm_status;
```

25. The procedure for receiving the packets from the WAN LINK CC3i is found in Step 1 and Step 2 of CC3iAPi.C. See, for example, Code Fragment 2 of Exhibit O (pages 16 - 17):

```
/*
      ** Step 1 : check if any data present.
      */
      if ( !(fd->status & USED) )
            {
            /* No frame, enable watchdog */
            check_cc3i_resync(iocb, inuse_port, NO_FRAMES);
            return(info);
            }

      /*
      ** Step 2 : gather length and verify frame is received completely.
      */
      iocb->next_rxout = iocb->rxout;
      discard = FALSE;
      frm_status = fd->status;
      if ((frm_status & EOFRM) && fd->count)
            {
            datacnt = fd->count;    /* add to total count */
            if (++iocb->next_rxout >= MAXDESC)
                  iocb->next_rxout = 0;
            }
      else  {
            datacnt = 0;
            do    {
                  if ( !(fd->status & EOFRM) &&
                              (fd->count != iocb->rxbufsize))
                        {
                        iocb->invalid_frames++;
                        discard = TRUE;
                        datacnt = 0
```

```
}
```

26. The program FECHW.C (Exhibit N) includes the instruction "Adapter->RxDescRing[Priority].AllocSize = ((sizeof(SF_RX_DESC) * SF_NUMBER_OF_RX_DESC)+ SF_DESC_ALIGN)", for initializing the memory. See, for example, Code Fragment 2 of Exhibit N (page 3):

```
[/*************************************************************

 * Function:     InitializeReceiveDesc                        *
 * Description:  Initializes Receive descriptors.             *
 * Returns:      TRUE, if successful.                         *
 *               FALSE, if failure.                           *
 *************************************************************/
PRIVATE BOOL InitializeReceiveDesc (Adapter, Priority)
      PSF_ADAPTER Adapter;
    UINT Priority;
{
UINT ii;

      if (Priority >= SF_NUMBER_OF_RX_QUEUES)
            return(TRUE); // Do not allocate this descriptor ring

      // Allocate memory to hold the receive descriptors (non-cached).
      Adapter->RxDescRing[Priority].AllocSize =
            ((sizeof(SF_RX_DESC) * SF_NUMBER_OF_RX_DESC)
            + SF_DESC_ALIGN);

    Adapter->RxDescRing[Priority].AllocVa =
            (N32) malloc (Adapter->RxDescRing[Priority].AllocSize);

      Adapter->RxDescRing[Priority].AllocPa =
            Adapter->RxDescRing[Priority].AllocVa;

      if (!Adapter->RxDescRing[Priority].AllocVa)
            {
            log_event ("<Error>: Insufficient Memory for Rx
            Descriptors");
        return FALSE;
            }

      memset ((VOID *)Adapter->RxDescRing[Priority].AllocVa, 0,
            Adapter->RxDescRing[Priority].AllocSize);
```

I. **A stripper for stripping header information associated with a protocol of the first network link (Claims 12, 34)**

27. As shown in Exhibits O and Q, the code includes instructions for "stripping header information associated with a protocol of the first network link", as recited in claim 12. Exhibits O and Q also contain instructions for features in claim 34, which, though of different scope from claim 12, recites similar elements and was rejected under the same rationale. CC3iAPi.C (Exhibit O), which, as indicated above, implements the WAN link driver CC3i, contains instructions to make calls to a function "strip_wanhdr" that operates to delineate or strip WAN protocol information from a received data packet. See, for example, Code Fragment 3 of Exhibit O (pages 17-18) :

```
  fd    = &iocb->rfd[iocb->rxout];
  info->frame_p = 0;

if(!(frm_status & CRCE)) /* 5.0 */
      strip_wanhdr(fd->datap, datacnt, info);
else
    {
    info->frame_p = fd->datap;
    info->frame_size = datacnt + FCS_LENGTH;
    }
```

J. **Stripping checksum information associated with a protocol of the first network link (Claim 14)**

28. As shown in Exhibit O, the code includes instructions for "stripping checksum information associated with a protocol of the first network link", as recited in claim 14. The instructions of CC3iAPi.C (Exhibit O), which were cited in paragraph 27 above to show stripping of data packets, also show stripping of Frame Check Sum (FCS) information. See, for example, Paragraph 27 for Code Fragment 1 of Exhibit O (pages 17-18).

K. **An adder for adding header information associated with a protocol of the second network link (Claims 12, 34)**

29. As shown in Exhibit P, the code includes instructions for "adding header information associated with a protocol of the second network link", as recited in claim 12. Exhibit P also contains instructions for features in claim 34, which, though of different scope from claim 12, recites similar elements and was rejected under the same rationale. Exhibit P consists of portions of a software program entitled DECAP.C, which contains calls to packet converter code for transforming header information into Ethernet header information. As noted above, the FEC driver, which is a Fast Ethernet driver, is an embodiment of the second network link. See, for example, the description of the DECAP.C program in Code Fragment 1 of Exhibit P (page 1):

```
/* Description:
                                        */
/*
                                                    */
/* Contains all functions pertaining decapsulating WAN header & making  */
/* the packet look like an EThernet frame.
            */
/*
```

30. DECAP.C also contains instructions for attaching a MAC header which will then be converted to a format suitable to the Ethernet FEC driver. See, for example, Code Fragment 2 of Exhibit P (pages 23-24):

```
/*
** function builds the proper protocol frame, adding the Ethernet MAC header.
*/
PRIVATE VOID build_x25_etpkt(dp, svc)
      N8 *dp;
      SVC *svc;
      {
      N16 length;
      N16 etype;

      wmac->xfptr = 0;
      HTON16 ((TRN16 *) &machdr[4], svc->vc);

      etype = svc->protocol;

      /* first check if this is a multiprotocol VC */
      if (etype == X25_MULTI_PROTOCOL)
            {
```

```
            etype = x25_guess_protocol(dp);
            if ( !etype )
                  return;
            }
      else if ((etype == X25_BAY_QLLC) || (etype == X25_CISCO_QLLC))
            {
            SAVE_WAN_HDR(dp+2);
            --dp;
            memcpy(dp, sna_llchdr, 3);
            length = wmac->wanfsize - wmac->hdrlen + 17;
            PUT_BE_16((TRN16 *) (dp - 2), length);
            wmac->xfptr = dp - 14;
            memcpy(wmac->xfptr, machdr, 12);
            }
      else if (etype == X25_SNA)
            {
            SAVE_WAN_HDR(dp);
            dp -= 3;
            memcpy(dp, sna_llchdr, 3);
            length = wmac->wanfsize - wmac->hdrlen + 17;
            PUT_BE_16((TRN16 *) (dp - 2), length);
            wmac->xfptr = dp - 14;
            memcpy(wmac->xfptr, machdr, 12);
            }
      else if (etype == X25_OSI)
            {
            dp -= 3;
            memcpy(dp, osi_llchdr, 3);
            length = wmac->wanfsize - wmac->hdrlen + 17;
            PUT_BE_16((TRN16 *) (dp - 2), length);
            wmac->xfptr = dp - 14;
            memcpy(wmac->xfptr, machdr, 12);
            }
      else if (etype == X25_BAY_PTOP)
            {
            SAVE_WAN_HDR(dp);
            wmac->xfptr = dp;
            }
      else if (etype != NULL_PID)
            {
            SAVE_WAN_HDR(dp);
            HTON16((TRN16 *)(dp-2), etype);
            wmac->xfptr = dp - 14;
            memcpy(wmac->xfptr, machdr, 12);
            }
      }

/*
```

31. DECAP.C also contains instructions for transforming the MAC headers into a format suitable to the Ethernet FEC driver. See, for example, Code Fragment 3 of Exhibit P (pages 33-34):

```
/*
** function strips out WAN headers, transforms MAC headers and gets
** encapsulated LAN data for upper layer functions to work on.
*/
EXPORT  VOID strip_wanhdr(datap, length, frame_info)
      N8 *datap;
      UINT length;
      XMIB_FRAME_INFO *frame_info;
      {
      /* PROTO_PARSER pfn; */
      TRN16 *len;

      if(datap==0) /* 5.0 */
                  {
                  frame_info->frame_p = 0;
                  frame_info->frame_size = 0;
                  return;
                  }
```

L. **Adding checksum information associated with the protocol of the second network link (Claim 14):**

32. As shown in Exhibits N and O, the code includes instructions for "adding checksum information associated with the protocol of the second network link", as recited in Claim 14. CC3iAPi.C (Exhibit O), which, as indicated above, implements the WAN link driver CC3i, contains instructions, "info->real_frame_size = datacnt + FCS_LENGTH" and "info->mac_errors = info->crc_align_error = (frm_status & CRCE)", to add Frame Check Sum information and delimiters and to mark frames with bad check sums with indicators. See, for example, Code Fragment 4 of Exhibit O (page 18):

```
if (info->frame_p)
      {
      /* Add FCS + start & ending delimiters, 1 byte each  */

      info->real_frame_size = datacnt + FCS_LENGTH;
      /* info->frame_size = datacnt + length_pad - length_strip; */

      info->mac_errors =
            info->crc_align_error = (frm_status & CRCE);

      if (info->dlci_valid)
            xmib_setup_dlci_parms(info);
```

```
        else
              info->et_collisions = iocb->stats->rx_abort;

        /* call RMON stats collector */
        xmib_collect_wanstats(info);

        info->frame_time_ms = timer_get_uptime();
        ++iocb->processed_frames;
        }
```

33. The program FECHW.C (Exhibit N) includes a header appending function to append the CRC or Check Sum for a data packet before the packet frame is transmitted into the network. See, for example, Code Fragment 4 of Exhibit N (page 19):

```
EXPORT BOOL XmtFrame (Adapter, TxBuffer, BufferLength, CRCAppend)
            PSF_ADAPTER Adapter;     // Adapter Pointer
      N8    *TxBuffer;               // Pointer to the frame to be xmted starting
                                     // from DA.
            UINT     BufferLength;   // Lengh of xmt buffer
      N8    CRCAppend;               // if TRUE, CRC will be appended before xmting
```

34. FECHW.C (Exhibit N) also includes instructions for appending the CRC for a data packet. See, for example, Code Fragment 1 of Exhibit N (page 2):

```
Adapter->TxDesc[Priority][ii]->u.DWORD0.CrcEn = 1;  // calc. CRC
```

See also Code Fragment 5 of Exhibit N (page 20):

```
TxDesc->u.DWORD0.CrcEn = (CRCAppend) ? 1 : 0;
```

M. <u>A performance analyzer for acquiring network performance data in response to the monitored packets communicated over the first network link (Claim 28)</u>

35. As shown in Exhibit K, Q and R, the code includes instructions for providing "a performance analyzer for acquiring network performance data in response to the monitored packets communicated over the first network link", as recited in claim 28. Exhibit K consists of portions of a software program entitled REALCOL.C, which, as indicated above at paragraph 15, contains network performance evaluation code. REALCOL.C contains code in which network performance data is collected into MIB's (Management Information Bases).

REALCOL.C contains an instruction, "EXPORT VOID mibmgr_collector(info)", which collects data. It also has instructions, "TRACE(MIBMGR_COLLECTOR)" and "fastpath_process_frame(info);." MIBMGR_COLLECTOR invokes fastpath_process_frame, which acquires data in accordance with the program FASTPATH.C, which is described below. See, for example, Code Fragment 1 of Exhibit K (page 6):

```
EXPORT VOID mibmgr_collector(info)
      XMIB_FRAME_INFO *info;
      {

        if ( mibmgr_inited == FALSE)
              return;
#if _QOS
      IMPORT BOOL xmib_qos_enabled;
      IMPORT BOOL isFrameIP;
      IMPORT N8 ipv4Tos;
      UINT qosifn;
      BOOL collect_qos_info = FALSE;

#endif
      TRACE(MIBMGR_COLLECTOR);

      if (info->channel)
            {
            /* setup channel interface indexes */
            info->ifn = info->channel->ch_ifn;
            info->lifn = info->channel->ch_lifn;
            }

      fastpath_process_frame(info);
      }
```

36. Exhibit Q consists of portions of a software program entitled FASTPATH.C, which contains code for acquiring network performance data. It also contains flow management and caching functions to speed up collector processing of the data. See, for example, Code Fragment 1 of Exhibit Q (pages 4-5):

```
EXPORT VOID fastpath_process_frame(XMIB_FRAME_INFO *frame_info)
      {
      PP_OUTPUT *pp_info;
      UINT i;
      CACHE_DATA *free_p;
      CACHE_DATA *p;
    N8 *packet;
IMPORT BOOL mib_config_dsmon;
```

```
IMPORT BOOL mib_config_hcrt;
IMPORT BOOL mib_config_art;

      /* Parse the frame. */

    if (frame_info->ifn != MIRROR_IFN)
          {
          pp_process_frame(frame_info);
          pp_info = (PP_OUTPUT *) frame_info->pp_info;
          }
        else // Handle the mirror ifn 49
          {
          pp_info = (PP_OUTPUT *) frame_info->pp_info;
          pp_info->ifn = MIRROR_IFN;
          }
#if _CDP
        /* Do any CDP processing.
        ** GGN: For 8800, CDP Pkts are received on the
        ** Channels ranging from 2048 - 2303
        */
        if ( pp_info->is_cdp && ((frame_info->ifn < LOGICAL_IFN_SEED)
||
          ((frame_info->ifn >= CHIFN_SEED) &&
              (frame_info->ifn < CHIFN_DTE_SEED)))
          )
              col_process_cdp(frame_info);
#endif // _CDP
      /* Special fastpath processing for udp and tcp frames only. */

      if ( !(pp_info->is_tcp || pp_info->is_udp) )
          {
          col_process_rmon2_output(frame_info);
          return;
          }
```

37. Exhibit R consists of portions of a software program entitled COLETST.C, which is invoked by FASTPATH.C for performing the data collection. COLETST.C has an instruction, "collect_wanstats", that collects statistics from the data communicated over the WAN link driver CC3i (the first network link). See, for example, Code Fragment 1 from Exhibit R (page 9):

```
/*
**    collect_wanstats()
**    This is the RMON-MIB WAN (ether) Stats group collection handler.
*/

PRIVATE VOID collect_wanstats(es, frame_ptr, frame_size, frame_status,
      frame_time, frame_id)
EtherStatsEntry *es;
N8 *frame_ptr;
UINT frame_size;
```

```
N32 frame_status;
N32 frame_time;
N32 frame_id;
{
#ifdef _WAN
        N64 *p;
      ETHERSTATS_ENTRY *Es;
      XMIB_FRAME_INFO *info;

      if (frame_time != frame_time);       /* these args aren't used */
      if (frame_id != frame_id);

      /*
      ** Octets & Packets are incremented by frame count amount, since
      ** in netflow the frame count may be more than 1, which is default
      ** for all other interfaces.
      */
      es->Octets += mibcol_curr_octets;
      es->Pkts += mibcol_curr_pkts;
```

N. Software run time-stamp

38. Exhibit G is a printout of the output of a software program entitled LINKTIME.C, which operates as a time-stamp, evidencing the execution of the software of Exhibits I – R prior to July 30, 2002. The date itself has been blocked to preserve its confidentiality in accordance with MPEP 715.07.II, but we declare that the time-stamp date was prior to July 30, 2002.

39. The probe and intrusion detection method disclosed in the computer software code shown in Exhibits I - R form the subject of the above-identified application. As can be see from the above, substantially all of the features set forth in claims 3-4, 8-12, 13-14, 22-23, 28, and 32-34 of the above-identified application are also described in the computer software code and printout of the system implementation shown in Exhibits G and I - R. In our opinion, the computer software code shown in Exhibits I - R and the printout of the output of the LINKTIME.C program shown in Exhibit G, which are dated prior to July 30, 2002, conclusively demonstrate the conception of the claimed invention prior to July 30, 2002, and acts supporting such conception in the U.S.

II. DILIGENCE FROM JULY 30, 2002 TO AUGUST 9, 2002

40. Prior to July 30, 2002, a patent application drafting project was started for this invention. The project resulted in the filing of the Provisional on August 9, 2002. The law firm handling patent matters on behalf of NetScout assigned Docket No. FSD-004 to the Provisional. On and after July 30, 2002 and prior to August 9, 2002 (the filing date for the Provisional), a draft of the Provisional was being revised.

41. As evidence of such work, attached hereto as Exhibit H is a copy of a portion of the law firm's invoice to assignee NetScout showing the attorney hours spent on reviewing and revising the draft Provisional. As evidenced by the attached copy of selected pages of the invoice, on July 25 and 26, 2002, 11.50 hours of attorney time were spent on matters on behalf of the assignee NetScout including a review and revision of the draft application for FSD-004. Thereafter, 4.25 hours of attorney time were spent on July 30, 2002 reviewing correspondence from Danny Lobo and conferring with Danny Lobo about the draft Provisional. Further, on Friday, August 2, 2002, 0.50 hours of attorney time were spent reviewing a document from Danny Lobo. After a break for the weekend, August 3 - 4, 2002, on Monday, August 5, 2002, an additional 4.00 hours of attorney time were spent reviewing Danny Lobo's document and revising the application. On August 6, 2002, 0.50 hours of attorney time were spent reviewing the application and, on August 7, 2002, 2.00 additional hours of attorney time were spent reviewing and revising the draft Provisional, which was filed two days later.

42. Clearly, the hours that Danny Lobo spent in reviewing versions of the draft Provisional in that period would be in addition to those hours of attorney time that were documented in Exhibits H.

43. In our opinion, the copy of a portion of the law firm's invoice to NetScout that is attached hereto as Exhibit H and that shows significant work performed in drafting the provisional patent application upon which the instant application is based in the period between

July 30, 2002 and August 9, 2002 conclusively demonstrates that we were diligent in achieving reduction to practice of the invention after July 30, 2002 but before August 9, 2002.

44. We declare further that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further, that the statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patents issuing thereon.

Dated: 2/1/08

Dated: 2/5/08

Exhibit G

```
char nsagent_linktime[] = "                    linktime.c";
```

TESTA, HURWITZ & THIBEAULT, LLP

Exhibit H

Netscout Systems, Inc.

Invoice: 197095
Page 13

07/25/2002	[redacted]	FSD-004: review and revise draft; confer Martinson regarding draft	3.50	[redacted] Pat
07/25/2002	[redacted]	(FSD-004) Incorporated the comments from Mr. Lobel into the application. Updated the application and figures with Mr. Heffan comments. Drafted email to Mr. Lobel re: publication.	4.25	[redacted] Pat
07/26/2002	[redacted]	FSD-004: Review and revise application.	1.25	[redacted] Pat
[redacted]				Pat
07/26/2002	[redacted]	(FSD-004) Reviewed agent manual disclosure. Incorporated Mr. Frank's comments into the application. Office conference with Mr. Heffan re: application. Drafted email to Mr. Lobel re: draft application.	2.50	[redacted] Pat
[redacted]				
07/30/2002	[redacted]	FSD-004: review correspondence from Lobo; telephone conferences with Lobo to review application draft	1.50	[redacted] Pat
[redacted]				
07/30/2002	[redacted]	(FSD-004) Telephones conference with Mr. Heffan and Mr. Lobel re: draft	2.75	[redacted] Pat

Exhibit H

NetScout Systems, Inc.

Invoice: 198190
Page 23

FSD-004 Enhanced Computer Network Intrus

Date	Name	Description	Hours	Amount
08/02/2002	[redacted]	FSD-004: review document from Lobo	0.50	[redacted]
[redacted]				
08/05/2002	[redacted]	FSD-004: review draft document from Lobo; review and revise patent application	4.00	[redacted]
08/06/2002	[redacted]	FSD-004: review changes to application with Martinson	0.50	[redacted]
[redacted]				
08/07/2002	[redacted]	FSD-004: review and revise patent application per Lobo comments; review changes with Martinson	2.00	[redacted]
[redacted]				

EXHIBIT I

drvcfg.c

```
/*****************************************************************************
*       COPYRIGHT (c)      , FRONTIER SOFTWARE DEV. INC                        *
*                    ALL RIGHTS RESERVED                                      *
*                                                                             *
* Module Name : drvcfg.c                                                      *
* Component of: MPC Software                                                  *
* Programmer  : Anil Singhal                                                  *
*                                                                             *
* Description:                                                                *
*       Contains driver configuration info for all models.                    *
*       Must be compiled using appropriate MODEL_XXXX switch.                 *
*                                                                             *
* Comments:                                                                   *
*                                                                             *
* Revision History:                                                           *
*                                                                             *
* Vers. Date        Who Why                                                   *
*                                                                             *
*                                                                             *
*                                                                             *
*                                                                             *
*                                                                             *
*****************************************************************************/
```

EXHIBIT I

```
#elif (MODEL_9200 && !_IDS)
#define FEC     1

#elif (MODEL_9200 && _IDS)
#define FEC     1
#define CC3i    1
```

Code Fragment 1

Exhibit J

fecmain.c

```
/*****************************************************************************
*                                                                            *
*                                 W A R N I N G                              *
*                                 ~~~~~~~~~~~~~                              *
*                                Copyright (c)                               *
*                                 NetScout Systems, Inc.                     *
*                                   Westford, MA 01886                       *
*                                  All Rights Reserved                       *
*                                                                            *
* This software is part of Licensed material, which is the property of       *
* NetScout Systems, Inc.  Unauthorized use, duplication or distribution      *
* is strictly prohibited by Federal law.  No title to and ownership of       *
* this software is hereby transferred.                                       *
*                                                                            *
*****************************************************************************/
/*****************************************************************************
*                                                                            *
* Module Name : FecMain.C                                                    *
* Component of: Fast Ethernet driver using the Adaptec Starfire adapter      *
* Description:  Contains the driver entry point functions conforming to      *
*       'drvapi' as defined by FSD.                                          *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
* Revision History:                                                          *
*                                                                            *
* Vers. Date            Who     Why                                          *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*****************************************************************************/
```

EXHIBIT J

```
// Allocate memory for descriptors+ buffers for both Rx and Tx
if (!(AllocateAdapterMemory(Adapter)))
        return (EDRV_CONFIG_FAIL);

// Initialize some variables in the adapter structure
Adapter->AvailableTxDesc = SF_NUMBER_OF_TX_DESC;
Adapter->PhysicalSegmentThreshold = SF_PHYSICAL_SEGMENT_THRESHOLD;
Adapter->TxBuffersInUse = 0;
Adapter->TxBufferIndex = 0;
Adapter->DPCQueued = FALSE;
Adapter->ResetInProgress = FALSE;
Adapter->RxGfpNoResponseInt = FALSE;
```

Code Fragment 1

EXHIBIT J

```
/*************************************************************************
 * Function:    FecRecvFrame.                                            *
 * Description: Receive Packet Handler.                                  *
 *              If NO pkts are received then info->frame_size will be    *
 *              set to ZERO.                                             *
 *                                                                       *
 * Returns:     XMIB_FRAME_INFO struct.                                  *
 *************************************************************************/
EXPORT XMIB_FRAME_INFO  *FecRecvFrame (ifn)
```

Code Fragment 2

EXHIBIT J

```
    // Check if frame has arrived.
    // A frame can arrive on the High Priority Q or on the Low Prority Q.
    // If non-partial xfr mode is used, frames will arrive on LoPri Q only
    // In partial xfr mode, the header will arrive on HiPri Q and trailer
    // will arrive on LoPri Q.
    // Note:  Hi Pri Q <=> Q2   and   Lo Pri Q <=> Q1

// Check the Completion Q1.
// Note: In partial xfr mode, only 1 receive completion Q (i.e. Q1) is
// used.
SF_READ_REG(CompletionQ1ProducerIndex, &CompletionQ1ProducerReg);
SF_READ_REG(CompletionQ1ConsumerIndex, &CompletionQ1ConsumerReg);

RxComQProducerIndex = (N16)
    CompletionQ1ProducerReg.b.RxCompletionQ1ProducerIndex;
RxComQConsumerIndex = (N16)
    CompletionQ1ConsumerReg.b.RxCompletionQ1ConsumerIndex;

if (RxComQConsumerIndex != RxComQProducerIndex)
    { // Frame has arrived
```

Code Fragment 3

EXHIBIT K


```
#define RMON1_CHANGES_FOR_RMON2 1
#define MCAST_BCAST_FIX         1
/*****************************************************************************
*                                                                            *
*                                W A R N I N G                               *
*                                ~~~~~~~~~~~~~                               *
*                             Copyright (c)                                  *
*                              NetScout Systems, Inc.                        *
*                                Westford, MA 01886                          *
*                               All Rights Reserved                          *
*                                                                            *
* This software is part of Licensed material, which is the property of       *
* NetScout Systems, Inc.  Unauthorized use, duplication or distribution      *
* is strictly prohibited by Federal law.  No title to and ownership of       *
* this software is hereby transferred.                                       *
*                                                                            *
*****************************************************************************/
/*****************************************************************************
*                                                                            *
* File Name : realcol.c                                                      *
* Component of: MIB Manager                                                  *
* Programmer  : Anil Singhal                                                 *
*                                                                            *
* Description:                                                               *
*       Contains all top-level Collector functions for LICs.                 *
*       Includes all channel and filter related functions also               *
*                                                                            *
* Comments:                                                                  *
*                                                                            *
* Revision History:                                                          *
*                                                                            *
* Vers. Date        Who Why                                                  *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*****************************************************************************/
```

EXHIBIT K

```
EXPORT VOID mibmgr_collector(info)
        XMIB_FRAME_INFO *info;
        {

        if ( mibmgr_inited == FALSE)
                return;
#if _QOS
        IMPORT BOOL xmib_qos_enabled;
        IMPORT BOOL isFrameIP;
        IMPORT N8 ipv4Tos;
        UINT qosifn;
        BOOL collect_qos_info = FALSE;
#endif
        TRACE(MIBMGR_COLLECTOR);

        if (info->channel)
                {
                /* setup channel interface indexes */
                info->ifn = info->channel->ch_ifn;
                info->lifn = info->channel->ch_lifn;
                }

       fastpath_process_frame(info);
```

Code Fragment 1

```
#elif (_FEC) || (_LINKAGGR)
    /*
    ** For FEC:  do this if only Fast Ether Channel option is on
    ** For 8700, 87CC: do this only if Link Aggrgation is on
    */
    if ((info->mirr_ifn_enabled == TRUE) && (info->ifn > 2))
                {
                /* save info ptrs. */
                UINT vifn  = info->vifn;
```

Code Fragment 2

EXHIBIT K

```
                UINT lifn  = info->lifn;
                UINT lvifn = info->logical_vifn;

                info->vifn = info->lifn = info->logical_vifn = 0;
                CALL_PP_PROCESS(MIRROR_IFN);

                /* retrieve the saved info ptrs. */
                info->vifn = vifn;
                info->lifn = lifn;
                info->logical_vifn = lvifn;
                }
#endif // model 73xx, 8200, 8700, 87cc
```

Code Fragment 2
(Continued)

EXHIBIT L

colcapt.c

```
/*****************************************************************************
*                                                                            *
*                              W A R N I N G                                 *
*                              ~~~~~~~~~~~~~                                 *
*                        Copyright (c)                                       *
*                    Frontier Software Development Inc.                      *
*                          Tewksbury, MA 01876                               *
*                          All Rights Reserved                               *
*                                                                            *
* This software is part of Licensed material, which is the property of       *
* Frontier Software Development Inc.  Unauthorized use, duplication or       *
* distribution is strictly prohibited by Federal law.  No title to and       *
* ownership of this software is hereby transferred.                          *
*                                                                            *
*****************************************************************************/
/*****************************************************************************
*       COPYRIGHT (c)              FRONTIER SOFTWARE DEV. INC                *
*                    ALL RIGHTS RESERVED                                     *
*                                                                            *
* Module Name : colcapt.c                                                    *
* Component of: NETscout LIC Software                                        *
* Programmer  : Rajeev Nadkarni, Anil Singhal                                *
*                                                                            *
* Description:  Contains all data and functions pertaining to the            *
*               RMON-MIB data capture group:                                 *
*                                                                            *
*                       - col_activate_capture()                             *
*                       - deactivate_capture()                               *
*                       - collect_capture()                                  *
*                       - retrieve_capture()                                 *
*                       - write_capture()                                    *
*                                                                            *
* Comments:                                                                  *
*                                                                            *
* Revision History:                                                          *
*                                                                            *
* Vers. Date        Who  Why                                                 *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*                                                                            *
*****************************************************************************/
```

EXHIBIT L

```
/*
** Capture related structures.
** All Packet indices are one-based.
*/

typedef struct
        {
        CaptureBufferEntry_HDR  hdr;
        TRACE_BUCKET_HDR *first_bucket; /* first logical bucket ptr     */
        TRACE_BUCKET_HDR *curr_bucket;  /* current used bucket ptr      */
        TRACE_BUCKET_HDR *ret_bucket;   /* bucket ptr, used in retrieve */
        N32              count_bucket;  /* no. of buckets in list       */
        N32              packet_index;  /* last packet in the buffer    */
        N32              capture_start_time;
                                        /* time (ms) capture started    */
        N32              max_bufsize;   /* Max buffer size allowed      */
    N32       bucket_size;
        /* These fields are used for allocate/commit/release operations. */
        TRACE_BUCKET_HDR *new_first_bucket;
        N32 new_bucket_count;
        }  CAPTURE_ROOT;
```

Code Fragment 1

EXHIBIT L

```
EXPORT BOOL col_activate_capture(handler_id, max_entry)
        UINT handler_id;
        UINT max_entry;
        {
        CAPTURE_ROOT *cr;

        TRACE(COL_ACTIVATE_CAPTURE);

        /* Allocate a CAPTURE root database. */
        if ((cr = malloc(sizeof(CAPTURE_ROOT))) == 0)
                return(FALSE);

        /*
        ** Invalidate all entries to force OP_CONFIG_COLLECTION to be done
        */
        memset(cr, 0, sizeof(CAPTURE_ROOT));
```


Code Fragment 2

```
    cr->hdr.operation = OP_CONFIG_COLLECTION;
    cr->packet_index = 1;
    cr->max_bufsize = ((N32) max_entry) << 10;

    /* Set pointers to collection and retrieval handler functions. */

    handler_list[handler_id].collector = collect_capture;
    handler_list[handler_id].retriever = retrieve_capture;
    handler_list[handler_id].writer    = write_capture;
    handler_list[handler_id].deactivate = deactivate_capture;
    handler_list[handler_id].stats_ptr = cr;

    return(TRUE);
    }
```

Code Fragment 2
(Continued)

```
PRIVATE VOID collect_capture(cr, frame_ptr, frame_size,
                             frame_status, frame_time, frame_id)
        CAPTURE_ROOT *cr;
        N8  *frame_ptr;
        UINT frame_size;
        N32 frame_status;
        N32 frame_time;
        N32 frame_id;
        {
        N32 pkt_time;
        CaptureBufferEntry *p;
        WAN_MACHDR *wmac = 0;
        ATM_HDR *AtmHdrPtr = 0;
        VLAN_INFO *vlan_info = 0;
        BOOL wanhdr_enabled;
        BOOL PassAtmHeader;
        BOOL PassVlanHdr;
        N8 *dp;
        UINT n, n1;
        N16 size;

       TRACE(COLLECT_CAPTURE);
```

Code Fragment 3

EXHIBIT L

```
if (cr->hdr.operation != OP_START_COLLECTION)
        return;             /* Nothing to do, if capture not on */

wanhdr_enabled = PassAtmHeader = PassVlanHdr = FALSE;
```

Code Fragment 3
(Continued)

EXHIBIT L

```
/* Now save the buffer in the bucket */
if ( (p = (CaptureBufferEntry *) capt_get_bufp(cr->curr_bucket)) == 0 )
        {
        log_event("collect_capture: NULL buffer ptr provided !\n");
        return;
        }

p = (CaptureBufferEntry *) ((N8 *) p + cr->curr_bucket->space_used);
```

Code Fragment 4

```
TRACE(RETRIEVE_CAPTURE);

if (key_len != key_len);        /* unreferenced argument */

if (key_type == KEY_TYPE_CAPTURE_HDR)
        return(&cr->hdr);

if ((key_type != KEY_TYPE_CAPTURE_DATA) &&
        (key_type != KEY_TYPE_FAST_CAPTURE_DATA))
        {
        log_event("retrieve_capture: invalid key type %#0.8lx",
                                                        key_type);
        return(0);
        }

/* check if capture has started ? */

if (cr->hdr.CapturedPackets == 0)
        return(0);

if (get_next)
        getnext_id(cr, key_ptr);

id = GET_BE_32((TRN32 *) key_ptr);

if (key_type == KEY_TYPE_FAST_CAPTURE_DATA)
        return( fast_retrieve_capture(cr, id) );
```

Code Fragment 5

EXHIBIT M


```
#define  LUFTHANSA  1
#define MIB_LAST_REQ_OK_FIX 1
#define  OSI_ADDRESS_MAP_SUPPORT  1
/*****************************************************************************
 *
 *                           W A R N I N G                                   *
 *                           ~~~~~~~~~~~~~                                   *
 *                           Copyright (c)                                   *
 *                        NetScout Systems, Inc.                             *
 *                         Westford, MA 01886                                *
 *                         All Rights Reserved                               *
 *                                                                           *
 * This software is part of Licensed material, which is the property of      *
 * NetScout Systems, Inc.  Unauthorized use, duplication or distribution     *
 * is strictly prohibited by Federal law.  No title to and ownership of      *
 * this software is hereby transferred.                                      *
 *                                                                           *
 *****************************************************************************
/****************************************************************************/
 *****************************************************************************
 * Module Name : retpdir.c                                                   *
 * Component of: MIB Manager                                                 *
 * Programmer  : Anil Singhal                                                *
 *                                                                           *
 * Description:                                                              *
 *   Retrieval routines for the RMON2 protocol directory group.              *
 *                                                                           *
 * Comments:                                                                 *
 *                                                                           *
 * Revision History:                                                         *
 *                                                                           *
 * Vers. Date          Who Why                                               *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
```

EXHIBIT M

```
case HANDLER_RMON_CAPTURE:
        bp = (BufferControlEntry *) p;
        bp->packet = calloc(PACKET_BUF_SIZE,1);
        if (bp->packet == 0)
                {
                mib_control_del(p);
                return(0);
                }
        ret_init_capture(p);
        break;

case HANDLER_RMON_EVENT:
        ep = (EventEntry *) p;
        ep->Community = 0;
        ep->Description = calloc(sizeof(N32)
                                + MAX_EVENT_DESC_LEN+1,1);
        if (ep->Description != 0)
                ep->Community
                        = calloc(1,sizeof(N32)
                                + MAX_EVENT_COMMUNITY_LEN+1);
        if ( (ep->Description == 0) || (ep->Community == 0) )
                {
                mib_control_del(p);
                return(0);
                }
        ret_init_event(p);
        break;
```

Code Fragment 1

EXHIBIT N

fechw.c

```
/*****************************************************************************
*       COPYRIGHT (c)      , NetScout Systems Inc.                              *
*                    ALL RIGHTS RESERVED                                       *
*                                                                              *
* Module Name : FecHw.C                                                        *
* Component of: Fast Ethernet driver using the Adaptec Starfire adapter       *
* Description:  Contains hardware specific funtions that are needed to:       *
*       o Allocate Memory for Receive/transmit                                 *
*       o Program the Starfire CHIP.                                           *
*       It does not contain code specific to the PHY (e.g. for                 *
*       auto-negotiation)                                                      *
*                                                                              *
* Compilation:  - To be compiled with DB=P and pragma=pack options             *
*       - The transmit logic can be disabled by undef'ing                      *
*       TRANSMIT (in FECmain.h) for Debug purposes                             *
*                                                                              *
* Tunable parameters: - # of receive Buffers (set to ??? in )                  *
*       - Size of Rx Buffers set to ?? (in ??.h)                               *
*       - PCI Latency timer (set to ?? clks in ??.h)                           *
*                                                                              *
* Debug options:       - Debug messages for individual driver entry fns        *
*       can be turned ON/OFF (via flags in FECmain.c)                          *
*       Note: While making an official release define                          *
*       "RELEASE" in FECmain.h  to turn off all DEBUG                          *
*       messages.                                                              *
*                                                                              *
* Revision History:                                                            *
*                                                                              *
* Vers. Date        Who  Why                                                   *
*                                                                              *
*                                                                              *
*                                                                              *
*                                                                              *
*****************************************************************************/
```

EXHIBIT N

```
Adapter->TxDesc[Priority][ii]->u.DWORD0.CrcEn = 1;  // calc. CRC
```

Code Fragment 1

EXHIBIT N

```
/*****************************************************************************
 * Function:     InitializeReceiveDesc                                       *
 * Description:  Initializes Receive descriptors.                            *
 * Returns:      TRUE, if successful.                                        *
 *               FALSE, if failure.                                          *
 *****************************************************************************/
PRIVATE BOOL InitializeReceiveDesc (Adapter, Priority)
        PSF_ADAPTER Adapter;
    UINT Priority;
{
UINT ii;

        if (Priority >= SF_NUMBER_OF_RX_QUEUES)
                return(TRUE); // Do not allocate this descriptor ring

        // Allocate memory to hold the receive descriptors (non-cached).
        Adapter->RxDescRing[Priority].AllocSize =
                ((sizeof(SF_RX_DESC) * SF_NUMBER_OF_RX_DESC)
                + SF_DESC_ALIGN);

   Adapter->RxDescRing[Priority].AllocVa =
          (N32) malloc (Adapter->RxDescRing[Priority].AllocSize);

       Adapter->RxDescRing[Priority].AllocPa =
           Adapter->RxDescRing[Priority].AllocVa;

       if (!Adapter->RxDescRing[Priority].AllocVa)
                {
                log_event ("<Error>: Insufficient Memory for  Rx Descriptors");
       return FALSE;
                }

       memset ((VOID *)Adapter->RxDescRing[Priority].AllocVa, 0,
           Adapter->RxDescRing[Priority].AllocSize);
```

Code Fragment 2

EXHIBIT N

```
/*****************************************************************************
 * Function:    AllocateAdapterMemory                                        *
 * Description: Allocates memory required for:                               *
 *      - Transmit descriptors and completion queue                          *
 *      - Transmit buffers (copy buffers for fragmented packets              *
 *      - Receive descriptors and completion queue                           *
 *      - Receive buffers and their flush buffers                            *
 *      - Structures to map xmt ring entries back to the packets             *
 *                                                                           *
 * Returns:     TRUE, if successful.                                         *
 *              FALSE, if failure.                                           *
 *****************************************************************************/
EXPORT BOOL AllocateAdapterMemory (Adapter)
        SF_ADAPTER      *Adapter;
{
UINT ii;

        //
        // Allocate Memory for XMT
        //
#ifdef ORIGINAL
UINT PoolBuffersNeeded;
N32  TmpAddress;

   // We need some NDIS_BUFFERs to describe memory that we
   // allocate (generally either to flush the buffer, or
   // because we need its physical address). To do this
   // we need a buffer pool, so we have to determine how
   // many buffers we need.
   PoolBuffersNeeded = SF_NUMBER_OF_RX_DESC + SF_MAX_TX_BUFFERS;

   NdisAllocateBufferPool(
        &AllocStatus,
        &Adapter->FlushBufferPoolHandle,
        PoolBuffersNeeded);

   if (AllocStatus != NDIS_STATUS_SUCCESS) {
      return FALSE;
   }
#endif
```

Code Fragment 3

EXHIBIT N

```
EXPORT BOOL XmtFrame (Adapter, TxBuffer, BufferLength, CRCAppend)
       PSF_ADAPTER Adapter;     // Adapter Pointer
    N8 *TxBuffer;               // Pointer to the frame to be xmted starting
                                // from DA.
       UINT     BufferLength;   // Lengh of xmt buffer
    N8 CRCAppend;               // if TRUE, CRC will be appended before xmting
```

Code Fragment 4

EXHIBIT N

```
TxDesc->u.DWORD0.CrcEn = (CRCAppend) ? 1 : 0;
```

Code Fragment 5

EXHIBIT O

cc3iapi.c

```
#define USE_RESERVED_MEMORY     1
/*************************************************************************/
/*      Copyright (c)      , Frontier Software Development, Inc.         */
/*                           ALL RIGHTS RESERVED.                        */
/*                                                                       */
/* Module Name:          cc3iapi.c                                       */
/* Component of:         CC3i, HSSI driver API library.                  */
/* Programmer:           Rajeev Nadkarni.                                */
/*                                                                       */
/* Description:                                                          */
/*                                                                       */
/* Contains all functions pertaining to the HSSI device driver using     */
/* SDL Communications CC3i card.                                         */
/*                                                                       */
/* Comments:     Use DO_TEST switch while testing code on DOS platform.  */
/*               Always use interface number (ifn) 1 in Test mode        */
/*                                                                       */
/* Revision History:                                                     */
/*                                                                       */
/* Vers.  Date         Who       Why                                     */
/*                                                                       */
/*************************************************************************/
```

EXHIBIT O

```
/*
** Receive the next frame if any from the specified port.
** Note : In the current implementation, if a frame size > recv buffer size
**        and the frame is spread in more than one buffer than only the
**        first buffer will have proper data, while the remaining data
**        copied by upper layers will be garbage. !!
*/
EXPORT XMIB_FRAME_INFO *cc3i_recv_frame(ifn)
        UINT ifn;
        {
        XMIB_FRAME_INFO *info;
        RXDESCR *fd;
        IOCB *iocb;
        UINT datacnt;
        INT  inuse_port;
        BOOL discard;
        N8   frm_status;
```

Code Fragment 1

```
        /*
        ** Step 1 : check if any data present.
        */
        if ( !(fd->status & USED) )
                {
                /* No frame, enable watchdog */
                check_cc3i_resync(iocb, inuse_port, NO_FRAMES);
                return(info);
                }

        /*
        ** Step 2 : gather length and verify frame is received completely.
        */
        iocb->next_rxout = iocb->rxout;
        discard = FALSE;
        frm_status = fd->status;
        if ((frm_status & EOFRM) && fd->count)
                {
                datacnt = fd->count;   /* add to total count */
                if (++iocb->next_rxout >= MAXDESC)
```


Code Fragment 2

EXHIBIT O

```
                iocb->next_rxout = 0;
        }
else    {
        datacnt = 0;
        do      {
                if ( !(fd->status & EOFRM) &&
                        (fd->count != iocb->rxbufsize))
                        {
                        iocb->invalid_frames++;
                        discard = TRUE;
                        datacnt = 0;
                        }
```

Code Fragment 2
(Continued)

```
    fd   = &iocb->rfd[iocb->rxout];
    info->frame_p = 0;

if(!(frm_status & CRCE)) /* 5.0 */
        strip_wanhdr(fd->datap, datacnt, info);
else
    {
```

Code Fragment 3

```
info->frame_p = fd->datap;
info->frame_size = datacnt + FCS_LENGTH;
}
```

Code Fragment 3
(Continued)

```
if (info->frame_p)
        {
        /* Add FCS + start & ending delimiters, 1 byte each  */

        info->real_frame_size = datacnt + FCS_LENGTH;
        /* info->frame_size = datacnt + length_pad - length_strip; */

        info->mac_errors =
                info->crc_align_error = (frm_status & CRCE);

        if (info->dlci_valid)
                xmib_setup_dlci_parms(info);
        else
                info->et_collisions = iocb->stats->rx_abort;

        /* call RMON stats collector */
        xmib_collect_wanstats(info);

        info->frame_time_ms = timer_get_uptime();
        ++iocb->processed_frames;
        }
```

Code Fragment 4

EXHIBIT P

decap.c

```
#define DEBUG(x)
/*****************************************************************************/
/*      Copyright (c)     , Frontier Software Development, Inc.
*/
/*

                                                                        */

                                                                        */
/* Description:
                                                */
/*
                                                                        */
/* Contains all functions pertaining decapsulating WAN header & making  */
/* the packet look like an EThernet frame.
                */
/*
                                                                        */
```

Code Fragment 1

EXHIBIT P

```
/*
** function builds the proper protocol frame, adding the Ethernet MAC header.
*/
PRIVATE VOID build_x25_etpkt(dp, svc)
        N8 *dp;
        SVC *svc;
        {
        N16 length;
        N16 etype;

        wmac->xfptr = 0;
        HTON16 ((TRN16 *) &machdr[4], svc->vc);

        etype = svc->protocol;

        /* first check if this is a multiprotocol VC */
        if (etype == X25_MULTI_PROTOCOL)
                {
                etype = x25_guess_protocol(dp);
                if ( !etype )
                        return;
                }
        else if ((etype == X25_BAY_QLLC) || (etype == X25_CISCO_QLLC))
                {
                SAVE_WAN_HDR(dp+2);
                --dp;
                memcpy(dp, sna_llchdr, 3);
                length = wmac->wanfsize - wmac->hdrlen + 17;
                PUT_BE_16((TRN16 *) (dp - 2), length);
                wmac->xfptr = dp - 14;
                memcpy(wmac->xfptr, machdr, 12);
                }
        else if (etype == X25_SNA)
                {
                SAVE_WAN_HDR(dp);
                dp -= 3;
                memcpy(dp, sna_llchdr, 3);
                length = wmac->wanfsize - wmac->hdrlen + 17;
                PUT_BE_16((TRN16 *) (dp - 2), length);
                wmac->xfptr = dp - 14;
                memcpy(wmac->xfptr, machdr, 12);
                }
        else if (etype == X25_OSI)
                {
                dp -= 3;
                memcpy(dp, osi_llchdr, 3);
                length = wmac->wanfsize - wmac->hdrlen + 17;
                PUT_BE_16((TRN16 *) (dp - 2), length);
                wmac->xfptr = dp - 14;
```


Code Fragment 2

```
        memcpy(wmac->xfptr, machdr, 12);
        }
else if (etype == X25_BAY_PTOP)
        {
        SAVE_WAN_HDR(dp);
        wmac->xfptr = dp;
        }
else if (etype != NULL_PID)
        {
        SAVE_WAN_HDR(dp);
        HTON16((TRN16 *)(dp-2), etype);
        wmac->xfptr = dp - 14;
        memcpy(wmac->xfptr, machdr, 12);
        }
}
```

Code Fragment 2
(Continued)

EXHIBIT P

```
/*
** function strips out WAN headers, transforms MAC headers and gets
** encapsulated LAN data for upper layer functions to work on.
*/
EXPORT  VOID strip_wanhdr(datap, length, frame_info)
        N8 *datap;
        UINT length;
        XMIB_FRAME_INFO *frame_info;
        {
```

Code Fragment 3

```
/* PROTO_PARSER pfn; */
TRN16 *len;

if(datap==0) /* 5.0 */
                {
                frame_info->frame_p = 0;
                frame_info->frame_size = 0;
                return;
                }
```

Code Fragment 3
(Continued)

EXHIBIT Q

```
/*****************************************************************************
 *                                fastpath.c                                 *
 *                                                                           *
 *                                W A R N I N G                              *
 *                                ~~~~~~~~~~~~~                              *
 *                             Copyright (c)                                 *
 *                              NetScout Systems, Inc.                       *
 *                                Westford, MA 01886                         *
 *                               All Rights Reserved                         *
 *                                                                           *
 * This software is part of Licensed material, which is the property of      *
 * NetScout Systems, Inc.  Unauthorized use, duplication or distribution     *
 * is strictly prohibited by Federal law.  No title to and ownership of      *
 * this software is hereby transferred.                                      *
 *                                                                           *
 *****************************************************************************/
/*****************************************************************************
 *                                                                           *
 * File Name : fastpath.c                                                    *
 * Component of: MIB Collector                                               *
 * Programmer  : Danny Lobo                                                  *
 *                                                                           *
 * Description:                                                              *
 *       Contains flow managment and caching functions to speed up           *
 *       collector processing.                                               *
 *                                                                           *
 * Comments:                                                                 *
 *                                                                           *
 * Revision History:                                                         *
 *                                                                           *
 * Vers. Date            Who      Why                                        *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *****************************************************************************/
```

EXHIBIT Q

```
EXPORT VOID fastpath_process_frame(XMIB_FRAME_INFO *frame_info)
        {
        PP_OUTPUT *pp_info;
        UINT i;
        CACHE_DATA *free_p;
        CACHE_DATA *p;
    N8 *packet;
IMPORT BOOL mib_config_dsmon;
IMPORT BOOL mib_config_hcrt;
IMPORT BOOL mib_config_art;

        /* Parse the frame. */

   if (frame_info->ifn != MIRROR_IFN)
            {
            pp_process_frame(frame_info);
            pp_info = (PP_OUTPUT *) frame_info->pp_info;
            }
       else // Handle the mirror ifn 49
            {
```

Code Fragment 1

EXHIBIT Q

```
            pp_info = (PP_OUTPUT *) frame_info->pp_info;
            pp_info->ifn = MIRROR_IFN;
            }
#if _CDP
        /* Do any CDP processing.
        ** GGN: For 8800, CDP Pkts are received on the
        ** Channels ranging from 2048 - 2303
        */
        if ( pp_info->is_cdp && ((frame_info->ifn < LOGICAL_IFN_SEED) ||
                ((frame_info->ifn >= CHIFN_SEED) &&
                        (frame_info->ifn < CHIFN_DTE_SEED)))
            )
                col_process_cdp(frame_info);
#endif // _CDP
        /* Special fastpath processing for udp and tcp frames only. */

        if ( !(pp_info->is_tcp || pp_info->is_udp) )
                {
                col_process_rmon2_output(frame_info);
                return;
                }
```

Code Fragment 1 (continued)

EXHIBIT R

```
/************************************* coletst.c
 *****************************************************************************
 *                          W A R N I N G                                    *
 *                          ~~~~~~~~~~~~~                                    *
 *                     Copyright (c)                                         *
 *              Frontier Software Development Inc.                           *
 *                     Tewksbury, MA 01876                                   *
 *                     All Rights Reserved                                   *
 * This software is part of Licensed material, which is the property of     *
 * Frontier Software Development Inc.  Unauthorized use, duplication or     *
 * distribution is strictly prohibited by Federal law.  No title to and     *
 * ownership of this software is hereby transferred.                        *
 *                                                                           *
 *****************************************************************************
/****************************************************************************/
 *****************************************************************************
 *       COPYRIGHT (c)            FRONTIER SOFTWARE DEV. INC                 *
 *                ALL RIGHTS RESERVED                                        *
 *                                                                           *
 * Module Name : coletst.c                                                   *
 * Component of: NETscout LIC Software                                       *
 * Programmer  : Anil Singhal                                                *
 *                                                                           *
 * Description:  Contains all data and functions pertaining to the           *
 *               RMON-MIB etherStats group:                                  *
 *                                                                           *
 *                        - col_activate_etherstats()                        *
 *                        - deactivate_etherstats()                          *
 *                        - collect_etherstats()                             *
 *                        - retrieve_etherstats()                            *
 *                                                                           *
 * Comments:                                                                 *
 *                                                                           *
 * Revision History:                                                         *
 *                                                                           *
 * Vers. Date               Who       Why                                    *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                           *
 *                                                                          t*
 *                                                                           *
 *                                                                           *
 *                                                                           *
 ****************************************************************************/
```

EXHIBIT R

```
/*
**      collect_wanstats()
**      This is the RMON-MIB WAN (ether) Stats group collection handler.
*/

PRIVATE VOID collect_wanstats(es, frame_ptr, frame_size, frame_status,
        frame_time, frame_id)
EtherStatsEntry *es;
N8 *frame_ptr;
UINT frame_size;
N32 frame_status;
N32 frame_time;
N32 frame_id;
{
#ifdef _WAN
        N64 *p;
        ETHERSTATS_ENTRY *Es;
        XMIB_FRAME_INFO *info;

        if (frame_time != frame_time);  /* these args aren't used */
        if (frame_id != frame_id);

        /*
        ** Octets & Packets are incremented by frame count amount, since
        ** in netflow the frame count may be more than 1, which is default
        ** for all other interfaces.
        */
        es->Octets += mibcol_curr_octets;
        es->Pkts += mibcol_curr_pkts;
```

Code Fragment 1